TIES442 - Tasks - Week 4

Deadlines for handing in the tasks, and their effects on the awarded points:

- max 3 points: 14 April 2015 10:15 am
- max 2 points: 21 April 2015 10:15 am
- max 1 point: 30 June 2015 12:00 pm

Grading principles, maximum points for solution:

- completely empty or missing solution (by 30 June): 0p
- 1 or more tasks with no answer at all: 1-2p
- 2 or 3 tasks (from 1b, 2b, 2d) that are missing the descriptions of algorithm progress: 2p
- well reasoned solutions to all tasks, including descriptions of algorithm progress that are more or less correct (1b, 2b, 2d): 4p

Edit: fixed a typo, added the forgotten material hints, updated the task 2 image and clarified the task description in b) and d), clarified that task 3 description.

The tasks will appear latest 7 April 10 am. Otherwise, the deadlines will be moved forward as well.

This week, we will apply the search methods that we reviewed on the previous week for searching in state trees and -graphs. In the tasks we will define state spaces for the given problems using a *successor*-function: this function takes as parameter the current state and generates its every possible successor state. This way, a state tree is formed, and it may turn into a graph if it is possible to return to a previously visited state from some of its successor states.

Some hints for finding material on topic:

- Russell & Norvig chapter 3, if you have access to the book
- Poole & Mackworth chapter 3
- Stanford course, slides 2-6
- CMU course, chapter 3 slide
- search terms ‘AI search methods’, ‘uninformed search’, ‘heuristic search’, …

Task 1

Let us consider a classical problem: on a river bank, there are three cannibals and three adventurers. They need to cross the river using a boat, that can carry one or two persons at a time. If the cannibals outnumber the adventurers on either river bank, the adventurers will be eaten. How is it possible to carry everyone across the river safely and with the fewest number of river crossings?

a. Define the problem as a state space. What is the problem’s

   - initial state,
   - *successor*-function, that forms from the current state all possible successor states,
   - goal test function, that checks whether the current state is an acceptable final, or goal state,
   - path cost function, that determines the cost from the initial state to a goal state?

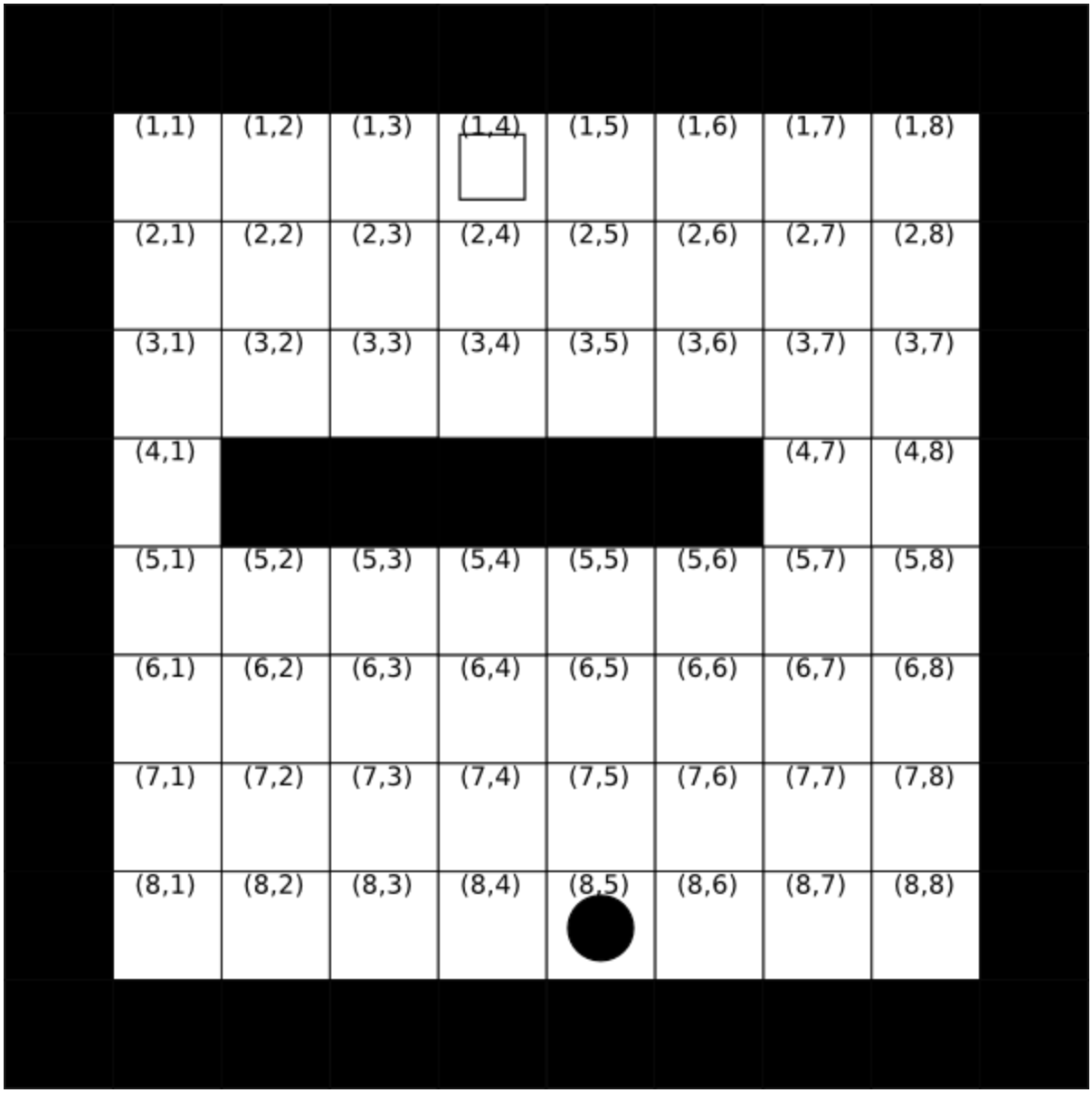
b. Select and describe a search method that finds the optimal solution, which means the path from

the initial state to a goal state with the smallest possible cost. Enumerate the states that the search method evaluates when it finds the optimal solution.

c. How would it be possible to avoid loops in the search tree, or how would it be possible to identify the states that have already been visited ?

Task 2

Let us examine the path finding problem for the vacuum cleaner robot we have described earlier using the situation depicted below. The robot, represented by the circle, must find the shortest path to the cell identified by the rectangle, which could be for example the charging station of the robot. The robot may move up, down, left and right. It is not possible to enter the black cells; we may assume, that there is a function that enumerates all legal moves in the current state.



Robot's world

a. Define the problem as a search problem in the same way as in the first task. Familiarize yourself

with the Dijkstra pathfinding algorithm, if it is not yet familiar to you. From the point of view of this algorithm, what is the problem's

- initial state,
- *successor*-function,
- goal test function,
- path cost function?

b. Describe, how pathfinding proceeds according to Dijkstra's algoritm. Enumerate the states that are visited and the cost of each state. Use the coordinates presented in the image as state labels.

c. Next, familiarize yourself with the A^* pathfinding algorithm (A-star). How does the definition of the search problem differ from the Dijkstra case? The algorithm requires a heuristic for estimating the length of the remaining path. Use the direct distance calculated using the Pythagoras theorem as a heuristic.

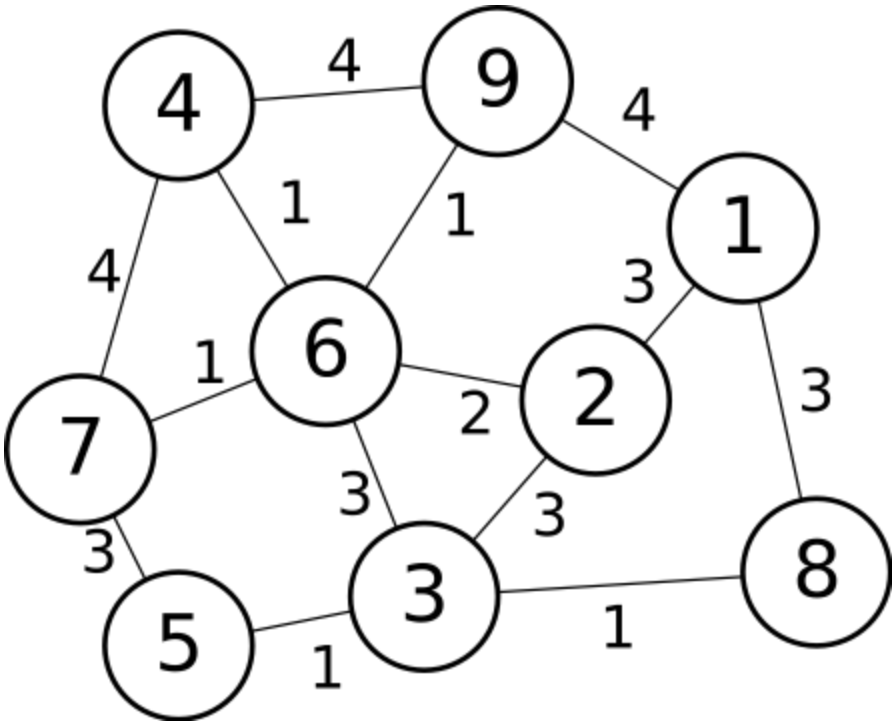
d. Describe, how pathfinding proceeds according to A^*-algorithm. Enumerate the states that are visited and the estimated cost at each state (using the heuristic), in the same way as in b).

e. In what way the A^* algorithm is better, and why?

f. How could the algorithm be changed, if the problem were much bigger and there would be enough memory for storing maximum n states? What if the world is not completely observable, and the agent has only access to next legal moves and the direct distance to goal state?

Task 3

Define solving the traveling salesperson problem as a search in the graph shown below. The numbers next to the edges describe the length of the journey between the nodes in question. Start from node 6. What kinds of heuristics could you use to reduce the number of states to check? How could you simplify the search, if you do not need to find the best possible solution? Let us specify, that it is enough if the solution is a *Hamiltonian path*, which means that the found route has to pass exactly once through each node, but it does not need to start and end in the same node.



Weighted graph
